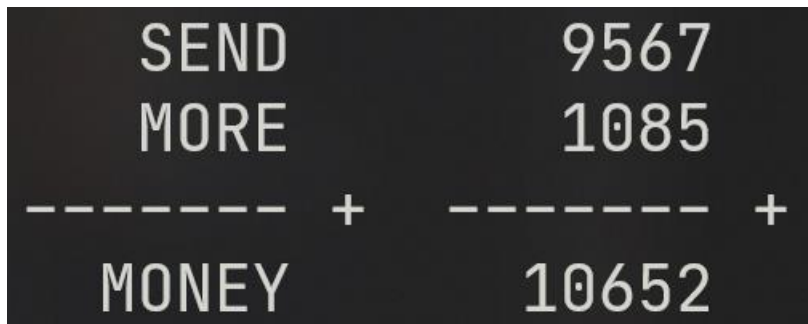


LAPORAN TUGAS KECIL PERTAMA

MATA KULIAH IF2211 STRATEGI ALGORITMA

ALGORITMA BRUTE FORCE



SEND		9567
MORE		1085
-----	+	-----
MONEY		10652

Nama/NIM:

Jeane Mikha Erwansyah/13519116

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

I. Algoritma *Brute Force*

1. Mengecek jumlah huruf unik dalam soal *cryptarithmic* jika lebih dari 10 soal tidak dapat dikerjakan. Sekaligus membuat kamus huruf dengan nilai awal 9.
2. Membuat daftar huruf-huruf pertama.
3. Melakukan permutasi angka 0-9 dengan panjang tertentu dengan angka tidak berulang. Melakukan permutasi rekursi untuk menghasilkan permutasi.
 - a. Terdapat 3 parameter, yaitu list char angka, hasil, dan sisa panjang.
 - b. Basis dari generator/fungsi adalah sisa panjang sama dengan 0.
 - c. Rekursinya adalah melakukan pengulangan (for loop) pengambilan char dari list char angka secara beurutan dan menambahkan char yang diambil ke hasil kemudian memanggil fungsi itu kembali dengan sisa panjang dikurangi 1.
 - d. Pada bagian rekursi dilakukan juga pengecekan keunikan char agar tidak diperoleh hasil dengan char yang berulang.
 - e. Jika sisa panjang sama dengan 0 maka generator akan mengeluarkan hasil permutasi dengan panjang yang diinginkan.
4. Mengecek apakah hasil permutasi memenuhi syarat, yaitu huruf pertama bukan angka 0.
5. Memetakan hasil permutasi ke kamus jika permutasi memungkinkan.
6. Mensubstitusi huruf dengan angka sesuai dengan kamus.
7. Menghitung operand. Jika nilai operand dan hasil sama, soal *cryptarithmic* memiliki jawaban.

II. Source Code Program

Berikut adalah *source code* program yang dibuat dengan bahasa pemrograman Python.

```
#!/usr/bin/env python3
# cryptarithmic.py
# 13519116 Jeane Mikha Erwansyah

from datetime import datetime

l1 = list()           # list operand dan hasil (huruf)
l2 = list()           # list operand dan hasil (angka)
allcrypt = list()     # list of list soal
solution = dict()     # dictionary dengan key huruf dan value angka
key = list()          # list key solution (huruf)
firstletters = list() # list huruf-huruf pertama
```

```

def replacechar(string, setofchar):
    # Fungsi mengembalikan string dengan setofchar yang sudah dihapus.
    string.upper()
    for char in setofchar:
        string = string.replace(char, '')
    return string

def parsefile():
    # Prosedur membaca file test.txt.
    # Hasil disimpan di list global allcrypt.
    # contoh hasil: [['NUMBER', 'NUMBER', 'PUZZLE'], ... ]
    global allcrypt
    ltemp = list()
    ltemp2 = list()
    f = open('../test/test.txt', 'r')
    for line in f:
        if line != '\n':
            temp = (replacechar(line.rstrip(), ['+', ' '])).upper()
            allcrypt.append(temp)
    f.close()

    i = 0
    # Membuat list soal cryptarithmic
    # yang berupa list operand dan hasil.
    # Membuang char '-' yang berulang
    while i < len(allcrypt):
        if allcrypt[i].find('-') == -1:
            ltemp.append(allcrypt[i])
        else:
            ltemp.append(allcrypt[i+1])
            ltemp2.append(list(ltemp))
            ltemp.clear()
            i += 1
    i += 1

    allcrypt.clear()
    allcrypt = ltemp2

```

```

def checkuniqueletters():
    # Fungsi mengecek jumlah huruf unik.
    # Fungsi menghasilkan nilai boolean
    # True apabila jumlah huruf unik kurang dari 10
    # dan False apabila lebih dari 10.
    global solution, key
    solution.clear()
    key.clear()
    for word in reversed(l1):
        for letter in range(len(word)):
            solution[word[letter]] = 9
            if len(solution) > 10:
                return False
    key = list(solution)
    return True

def permutations(numlist, l1, remainder):
    # Generator rekursif menghasilkan permutasi
    if (remainder == 0):
        yield tuple(l1)

    for i in range(10):
        l2 = l1 + numlist[i]
        if len(l2) == len(set(l2)):
            # Benar jika semua huruf/angka unik
            yield from permutations(numlist, l2, remainder - 1)

def listoffirstletters():
    # Prosedur membuat list huruf-huruf pertama.
    for keyindex in range(len(solution)):
        for element in range(len(l1)):
            if key[keyindex] == l1[element][0]:
                if key[keyindex] not in firstletters:
                    firstletters.append(key[keyindex])
                # key[keyindex] = key yang berupa huruf

def possible(sol):
    # Fungsi mengecek kemungkinan solusi permutasi.
    # Fungsi menghasilkan nilai boolean
    # True bila solusi permutasi mungkin
    # False jika tidak.
    for keyindex in range(len(solution)):
        if key[keyindex] in firstletters and sol[keyindex] == '0':
            return False
    return True

```

```

def substitute(l1):
    # Fungsi menghasilkan list string bilangan yang telah
    # diganti dari huruf.
    lsub = list()
    for element in l1:
        for word, letter in solution.items():
            element = element.replace(word, str(letter))
        lsub.append(element)
    return lsub

def solve():
    # Menyelesaikan cryptarithmic.
    # Jika tidak berhasil nilai -1 dihasilkan
    global solution, key, cases, firstletters, start, l2, end
    cases = 0
    firstletters.clear()
    start = datetime.now()
    listoffirstletters() # membuat list huruf-huruf pertama
    for sol in permutations(list('1023456789'), "", len(solution)):
        lefthand = 0
        righthand = 0
        if possible(sol):
            index = 0
            for i in sol: # mapping solusi ke dictionary
                solution[key[index]] = i
                index += 1
            l2 = substitute(l1)
            righthand = int(l2[len(l2)-1])
            for operand in range(len(l2)-1): # menghitung operand
                lefthand += int(l2[operand])
            cases += 1
            if lefthand == righthand:
                end = datetime.now()
                return

def transform(l1, l2):
    # Fungsi menghasilkan list list pasangan operand dan hasil
    l3 = list()
    l4 = list()
    for i in range(len(l1)):
        l3.append(l1[i])
        l3.append(l2[i])
        l4.append(list(l3[i*2:(i*2+2)]))
    return l4

```

```

def printsolution():
    # Prosedur mencetak solusi
    data = transform(l1,l2)
    width = max(len(word) for row in data for word in row) + 6
    i = 0
    for row in data:
        if i == len(data)-1:
            print('      ' + (width-4)* '-' + ' + ' + ' + (width-4)* '-' + ' +')
            print(''.join(word.rjust(width) for word in row))
        else:
            print(''.join(word.rjust(width) for word in row))
        i += 1

    print('Jumlah uji kasus: ' + str(cases) + ' kasus')
    print(f'Waktu eksekusi : {end - start}')

if __name__ == '__main__':
    parsefile()
    for q in range(len(allcrypt)):
        l1.clear()
        l1 = allcrypt[q]
        print('\n=>> Soal nomor ' + str(q+1))
        if checkuniqueletters():
            solve()
            lefthand = 0
            for i in range(len(l2)-1):
                lefthand += int(l2[i])
            if lefthand == int(l2[len(l2)-1]):
                printsolution()
            else:
                print('Tidak ada solusi.')
        else:
            print('Tidak ada solusi karena huruf unik lebih dari 10.')

```

III. Hasil Eksekusi Program

```
⇒ Soal nomor 1
  NUMBER      201689
  NUMBER      201689
  ----- + ----- +
  PUZZLE      403378
Jumlah uji kasus: 1016428 kasus
Waktu eksekusi : 0:00:25.093238
⇒ Soal nomor 2
  TILES       91542
  PUZZLES     3077542
  ----- + ----- +
  PICTURE     3169084
Jumlah uji kasus: 666828 kasus
Waktu eksekusi : 0:00:18.083800
⇒ Soal nomor 3
  CLOCK       90892
  TICK        6592
  TOCK        6892
  ----- + ----- +
  PLANET      104376
Jumlah uji kasus: 11225 kasus
Waktu eksekusi : 0:00:00.245517
⇒ Soal nomor 4
  COCA        8186
  COLA        8106
  ----- + ----- +
  OASIS       16292
Jumlah uji kasus: 7946 kasus
Waktu eksekusi : 0:00:02.347429
⇒ Soal nomor 5
  HERE        9454
  SHE         894
  ----- + ----- +
  COMES       10348
Jumlah uji kasus: 1058 kasus
Waktu eksekusi : 0:00:00.078765
```

(a)

```
⇒ Soal nomor 6
  DOUBLE      798064
  DOUBLE      798064
  TOIL        1936
  ----- + ----- +
  TROUBLE     1598064
Jumlah uji kasus: 180928 kasus
Waktu eksekusi : 0:00:04.000450
⇒ Soal nomor 7
  NO          87
  GUN         908
  NO          87
  ----- + ----- +
  HUNT        1082
Jumlah uji kasus: 1285 kasus
Waktu eksekusi : 0:00:00.350881
⇒ Soal nomor 8
  THREE       84611
  THREE       84611
  TWO         803
  TWO         803
  ONE         391
  ----- + ----- +
  ELEVEN      171219
Jumlah uji kasus: 199754 kasus
Waktu eksekusi : 0:00:05.731664
⇒ Soal nomor 9
  CROSS       96233
  ROADS       62513
  ----- + ----- +
  DANGER      158746
Jumlah uji kasus: 157029 kasus
Waktu eksekusi : 0:00:03.366757
⇒ Soal nomor 10
  MEMO        8485
  FROM        7358
  ----- + ----- +
  HOMER       15843
Jumlah uji kasus: 6452 kasus
Waktu eksekusi : 0:00:02.186040
```

(b)

Gambar 1. (a) dan (b) Tangkap Layar Hasil Eksekusi Program

IV. Pranala Kode Program

<https://github.com/jerwansyah/Stima-01>

Lampiran

Tabel 1. Tabel Pembantu Penilaian

<i>POIN</i>	<i>Ya</i>	<i>Tidak</i>
1. Program berhasil dikompilasi tanpa kesalahan (<i>no syntax error</i>).	✓	
2. Program berhasil berjalan.	✓	
3. Program dapat membaca file masukan dan menuliskan luaran.	✓	
4. Solusi <i>cryptarithmic</i> hanya benar untuk persoalan <i>cryptarithmic</i> dengan dua buah operand.		✓
5. Solusi <i>cryptarithmic</i> benar untuk persoalan <i>cryptarithmic</i> dengan lebih dari dua buah operand.	✓	